HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80528-9599

06-16-00

PATENT APPLICATION

ATTORNEY DOCKET NO. 10992563-1

A

JC830 U.S. PTO 00/15/00

IN THE U.S. PATENT AND TRADEMARK OFFICE
Patent Application Transmittal Letter

JC836 U.S. PTO 09/595036 06/15/00

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

SIR:

Transmitted herewith for filing under 37 CFR 1.53(b) is a(n): (X) Utility ( ) Design

(X) original patent application,

( ) continuation-in-part application

INVENTOR(S): **Ted Scott Rakel et al**

TITLE: **Method For Determining The DC Margin Of A Latch**

Enclosed are:

(X) The Declaration and Power of Attorney. (X) signed ( ) unsigned or partially signed

(X) 7 sheets of drawings (one set) ( ) Associate Power of Attorney

( ) Form PTO-1449 ( ) Information Disclosure Statement and Form PTO-1449

( ) Priority document(s) ( )(Other) _____ (fee $_____)

CLAIMS AS FILED BY OTHER THAN A SMALL ENTITY				
(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) TOTALS
TOTAL CLAIMS	20 — 20	0	X $18	$ 0
INDEPENDENT CLAIMS	3 — 3	0	X $78	$ 0
ANY MULTIPLE DEPENDENT CLAIMS	0		$260	$ 0
BASIC FEE: Design $310.00); Utility $690.00)				$ 690
TOTAL FILING FEE				$ 690
OTHER FEES				$
TOTAL CHARGES TO DEPOSIT ACCOUNT				$ 690

Charge $ 690 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16, 1.17,1.19, 1.20 and 1.21. A duplicate copy of this sheet is enclosed.

"Express Mail" label no. EL634172995US

Date of Deposit 6/15/00

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

By Linda C. Cunningham

Typed Name: **Linda C. Cunningham**

Respectfully submitted,

Ted Scott Rakel et al

By _____

Alexander J Neudeck

Attorney/Agent for Applicant(s)

Reg. No. 41,220

Date: 6-14-00

Telephone No.: **(970) 898-4931**

Rev 12/99 (TransNew)

- Attach as First Page to Transmitted Papers -

METHOD OF DETERMINING DC MARGIN OF A LATCH

Technical Field

The invention relates to electronic circuits. More particularly, the invention relates to simulation and determination of design parameters of an electronic circuit.

Background Art

A latch is a circuit element that maintains a particular state between state changing events, i.e., in response to a particular input, and is ubiquitous in digital sequential circuit designs. For example, as shown in Fig. 1, a typical latch **100** may include, inter alia, a forward inverter **101**, a feedback inverter **102**, an input terminal **103** and an output terminal **104**. The output voltage level, V_{OUT}, remains at a particular voltage level, i.e., either high or low, until an input signal, V_{IN}, is received at the input terminal **103**, at which time the state of the output may change depending on the nature of the input signal. For example, the state of the output **104** may change from a high state to a low state upon receipt of a logical high signal at the input **103**.

In order for the latch to operate properly, i.e., to change state upon receiving a particular input, the input signal levels to the latch must exceed certain thresholds with a sufficient margin. To this end, during a circuit design, it must be ensured that the input signal levels delivered through various signal paths to each of latches in the circuit under design meet the above input signal margin.

One of the ways to ensure satisfaction of the above input signal level requirement is to determine what is often referred to as the "DC margin" for each of the latches present in the circuit being designed.

The DC margin is a pair of values, the one margin and the zero margin. The one margin is the difference between the trip voltage (V_{trip}) of the forward inverter **101** of the latch **100** and the worst case pull-up input signal level that may be presented to the latch. The zero margin is the difference between the V_{trip} of the forward inverter **101** and the worst case pull-down input signal level that may be presented to the latch **100**. The trip voltage V_{trip} is defined as the equilibrium voltage level of the output voltage level and the input voltage level of the forward inverter. In order for a particular circuit design to be deemed acceptable, the DC margin must exceed a minimum margin according to a design guideline.

Unfortunately, heretofore, in order to determine the DC margin of a latch, every possible signal paths from each of the possible circuit elements that may drive the latch must be examined, requiring performance of simulations, using a simulation program, e.g., the SPICE™, for each of the possible signal paths. The prior attempts to determine the DC margin requires numerous simulations, each of which takes a long time to perform, and are thus inefficient and time consuming. This problem may be exacerbated if there are numerous latches in the particular circuit under design.

Thus, there is a need for more efficient method of determining DC margin of a latch, which does not require numerous simulations for every possible signal paths to the latch.

Summary of Invention

In accordance with the principles of the present invention, a method of determining a DC margin of a latch comprises performing a first simulation using a first simulation circuit to determine a trip voltage of a forward inverter of the latch, performing a second simulation using a second simulation circuit to determine a one margin of the latch, the second simulation circuit comprising a worst case pull-up signal path, and performing a third simulation using a third simulation circuit to determine a zero margin of the latch, the third simulation circuit comprising a worst case pull-down signal path .

In accordance with another aspect of the principles of the present invention, a computer program stored on a computer readable storage medium implements a method of determining a DC margin of a latch, and comprises a set of instructions for performing a first simulation using a first simulation circuit to determine a trip voltage of a forward inverter of the latch, performing a second simulation using a second simulation circuit to determine a one margin of the latch, the second simulation

circuit comprising a worst case pull-up signal path, and performing a third simulation using a third simulation circuit to determine a zero margin of the latch, the third simulation circuit comprising a worst case pull-down signal path .

In yet another aspect of the principles of the present invention, a simulation circuit for determining a DC margin of a latch comprises a latch portion representing the latch being simulated, the latch portion comprising a forward inverter and a feedback inverter, an input of the forward inverter being operably connected to an input of the latch portion, and an input of the feedback inverter being operably connected to an output of the latch portion, a driver portion representing a driver circuit element capable of supplying an input signal to the latch being simulated, and a pass path subcircuit configured to receive a drive signal from the driver portion, and configured to supply the drive signal to the input of the latch portion, the pass path subcircuit representing one or more pass circuit elements along a worst case signal path between the driver circuit element and the latch being simulated.

Description of Drawings

Features and advantages of the present invention will become apparent to those skilled in the art from the following description with reference to the drawings, in which:

Figure 1 is a logic diagram of showing the relevant portions of a conventional latch.

Figure 2 is a circuit diagram illustrative of an embodiment of a simulation model circuit of the a latch and the corresponding input signal path to the latch in accordance with the principles of the present invention;

Figure 2A is a circuit diagram illustrative of an embodiment of the path subcircuit shown in Fig. 2;

Figure 3 is flow diagram illustrative of an exemplary embodiment of the process of determining the DC margin of a latch in accordance with an embodiment of the principles of the present invention;

Figure 4 is a circuit diagram illustrative of an embodiment of a simulation circuit for determining the trip voltage of a forward inverter of a latch in accordance with the principles of the present invention;

Figure 5 is a circuit diagram illustrative of an embodiment of a simulation circuit for determining the one margin of a latch in accordance with the principles of the present invention; and

Figure 6 is a circuit diagram illustrative of an embodiment of a simulation circuit for determining the zero margin of a latch in accordance with the principles of the present invention.

Detailed Description of Preferred Embodiments

For simplicity and illustrative purposes, the principles of the present invention are described by referring mainly to an exemplar embodiment, particularly, with a specific exemplary implementations of various simulation circuits. However, one of ordinary skill in the art would readily recognize that the same principles are equally applicable to, and can be implemented in, other implementations and designs using any other equivalent simulation circuits, and that any such variation would be within such modifications that do not depart from the true spirit and scope of the present invention.

In accordance with the principles of the present invention, a DC margin of a latch of a circuit under design is determined by performing three simulations. A simulation is performed to find the trip voltage of the forwarding inverter of the latch. A second simulation is performed to find the one margin of the latch. Lastly, a third simulation is performed to find the zero margin of the latch.

During each of the simulations to find the one margin and the zero margin, the worst case input signal path from the various driver circuit elements and signal paths within the circuit under design is determined analytically by accumulating weighted resistance of each of the circuit elements along the signal paths. The weights assigned to the circuit elements are empirically determined based on the topology configuration of each of the circuit elements, e.g., the type circuit element, the signal being passed through the circuit element and whether a threshold voltage drop occurs between the drive circuit element and the pass circuit element.

In particular, Fig. 2 shows a simulation model circuit **200** representing a latch and its input signal path in accordance with an embodiment of the present invention. The simulation model circuit represents an equivalent circuit model of the actual circuit being simulated, and may not include every circuit elements present in the actual circuit being simulated. For example, the NFET Q1 **201** may represent more than one circuit element, e.g., a series of NFETs, of the actual circuit being simulated.

It should be understood by those familiar with circuit evaluation techniques that each of circuit elements shown in the model circuit **200** shown in Fig. 2 may be an equivalent circuit of, and represent, a number of circuit elements of the actual circuit under simulation.

The simulation model circuit comprises a latch model **100'**, which includes two complementary field effect transistor (FET) pairs, each representing the forward inverter and the feedback inverter of the latch. The complementary pair Q1 **201** and Q2 **202** together represent the forward inverter while the complementary pair Q3 **203** and Q4 **204** represent the feedback inverter.

The model circuit **200** further comprises a path subcircuit **207** which represent the various circuit elements along a signal path between the input **103** of the latch and a circuit element that may drive the input of the latch. The circuit element that drives the input of the latch is represented by the FETs Q5 **205** and Q6 **206**, each providing a pull-down input signal and a pull-up input signal, respectively, to the input of the latch (i.e., Vin **103**) through the path subcircuit **207**. As shown, when a logical low signal V_{INH} **208** is supplied to the gate of the PFET Q6 **206**, the PFET Q6 **206** is turned on, and thus Vin **103** is driven high. When a logical high signal V_{INL} **208** is supplied to the gate of the NFET Q5 **205**, the PFET Q5 **205** is turned on, and thus Vin **103** is driven low.

Fig. 2A shows the path subcircuit **207** in more detail. According to an embodiment of the present invention, the path subcircuit **207** comprises n/2 complementary pairs of FETs, Q_{S1} to Q_{Sn}, each of which can be configured to have variable size, and can be individually removed from the path subcircuit **207**, to simulate the worst case pull-up and pull-down signal paths. In a preferred embodiment of the present invention, the path subcircuit **207** comprises six (6) complementary pairs of FETs.

In an embodiment of the present invention, the worst case pull-up and pull-down signal paths are determined analytically by traversing through each of possible signal paths from the input of the latch to each of circuit elements that can drive the input of the latch, and identifying pass circuit elements along each of the signal paths, through which a signal may pass. For each of identified pass circuit elements, a weight is applied to its resistance, e.g., based on its length and width (L/W). The weights applied to the resistance of the pass circuit elements are empirically determined based on the topology configuration of each of the circuit elements, e.g., the pass circuit element type (e.g., whether the pass

circuit element is a single NFET, single PFET or a complementary pair of FETs), the signal being passed through the pass circuit element and whether a threshold voltage drop occurs between the drive circuit element and the pass circuit element.

The weighted resistance of the identified pass circuit elements along a particular signal path are added together to determine the total resistance of the particular signal path, and is compared to the total resistance similarly calculated for other signal paths to determine the worst case signal path which results in the worst degradation of signal level while passing through the signal path.

In a preferred embodiment of the present invention, the path subcircuit **207** is configured to include one or more of the FETs, Q_{S1} and Q_{Sn}, each included FET representing identified pass circuit elements. The sizes of the included FETs in the path subcircuit **207** are based on the sizes of the respective pass circuit elements. In an alternative embodiment, the path subcircuit **207** may simply be a resistor element having the weighted total resistance of the worst case signal path identified.

The inventive process of determining DC margin of a latch will now be described with references to FIGs. 4 through 6. In step **301**, a simulation, e.g., using SPICE™ which is known to those familiar with circuit design testing, is performed to determine the trip point voltage (V_{trip}) of the forward inverter of the latch. An exemplary embodiment of the simulation circuit **400** for the V_{trip} determination is shown in Fig. 4. In this example, the simulation circuit **400** includes the forward inverter portion of the latch, and comprises the complementary pair of FETs, Q1 **201** and Q2 **202**.

As shown, in the simulation circuit **400**, the input terminal Vin **103** and the output terminal Vout **104** of the forward inverter are connected together. The simulation comprises a DC analysis of the simulation circuit **400** to determine the settling voltage level at the input terminal Vin **103**, which is the same as the voltage level at the output terminal Vout **104**. Once the voltage levels at the input terminal and the output terminal settles to equal each other, th settling voltage is the trip point voltage (V_{trip}) of the forward inverter of the latch.

In step **302**, the worst case pull-up signal path and the path subcircuit **207** based on the worst case pull-up signal path are found as previously described. In particular, in an embodiment of the present invention, the worst case pull-up signal path is determined analytically by traversing through each of possible signal paths from the input of the latch to each of pull-up driver circuit elements that

can drive a pull-up signal to the input of the latch, and identifying pass circuit elements along each of the signal paths, through which a signal may pass. For each of identified pass circuit elements, a weight is applied to its resistance, e.g., based on its length and width (L/W). The weights applied to the resistance of the pass circuit elements are empirically determined based on the topology configuration of each of the circuit elements, e.g., the pass circuit element type (e.g., whether the pass circuit element is a single NFET, single PFET or a complementary pair of FETs), the signal being passed through the pass circuit element and whether a threshold voltage drop occurs between the pull-up driver circuit element and the pass circuit element.

The weighted resistance of the identified pass circuit elements along a particular signal path are added together to determine a cumulative resistance of the particular signal path, i.e., from the supply voltage V_{DD} through the pull-up driver circuit element and the pass circuit elements to the latch input. The cumulative resistence of each signal path is compared to the total resistance similarly calculated for other signal paths to determine the worst case pull-up signal path, i.e., the signal path with the highest cumulative resistance, which results in the worst degradation of signal level while passing through the signal path.

The pull-up driver of the identified worst case pull-up signal path is mapped to an equivalent canonical driver circuit Q6 **206** as shown in Fig. 5. The path subcircuit **207a** is constructed to include one or more of the FETs, Q_{S1} and Q_{Sn} (as shown in Fig. 2A), each included FET representing identified pass circuit elements of the worst case pull-up signal path. The sizes of the FETs included in the path subcircuit **207** are based on the sizes of the respective pass circuit elements.

In step **303**, a simulation is performed to determine the one margin ($Vmar_{(1)}$). An exemplary embodiment of the simulation circuit **500** for the $Vmar_{(1)}$ determination is shown in Fig. 5. The $Vmar_{(1)}$ simulation circuit **500** includes the forward inverter portion, i.e., the complementary pair of FETs Q1 and Q2, the NFET Q3, the PFET Q6 and the path subcircuit **207a** constructed in step **302** above. In this exemplary simulation circuit **500**, the FETs Q4 **204** and Q5 **205** shown in Fig. 2 are omitted. However, in an alternative embodiment, the FETs Q4 **204** and Q5 **205** may be included, and turned off.

The $Vmar_{(1)}$ simulation comprises a DC analysis of the simulation circuit **500** with the output Vout **104** initially set to logical high, i.e., set to V_{DD}, and determining the voltage level at the input terminal Vin **103**. If the voltage level at the input terminal Vin **103** exceeds the V_{trip}, and is sufficiently high to overcome the NFET Q3 **203**, the state of the output, Vout **104**, would switch from the high initial setting to a logical low. The one margin, $Vmar_{(1)}$, is the difference of the voltage level at Vin **103** and V_{trip}, i.e., $Vmar_{(1)} = Vin - V_{trip}$.

In step **304**, the worst case pull-down signal path and the path subcircuit **207** based on the worst case pull-down signal path are found as previously described. In particular, in an embodiment of the present invention, the worst case pull-down signal path is determined analytically by traversing through each of possible signal paths from the input of the latch to each of pull-down driver circuit elements that can drive a pull-down signal to the input of the latch, and identifying pass circuit elements along each of the signal paths, through which the pull-down signals may pass. For each of identified pass circuit elements, a weight is applied to its resistance, e.g., based on its length and width (L/W). The weights applied to the resistance of the pass circuit elements are empirically determined based on the topology configuration of each of the circuit elements, e.g., the pass circuit element type (e.g., whether the pass circuit element is a single NFET, single PFET or a complementary pair of FETs), the signal being passed through the pass circuit element and whether a threshold voltage drop occurs between the pull-down driver circuit element and the pass circuit element.

The weighted resistance of the identified pass circuit elements along a particular signal path are added together to determine a cumulative resistance of the particular signal path, i.e., from the ground (GND) through the pull-down driver circuit element and the pass circuit elements to the latch input. The cumulative resistence of each signal path is compared to the total resistance similarly calculated for other pull-down signal paths to determine the worst case pull-down signal path, i.e., the signal path with the highest cumulative resistance.

The pull-down driver of the identified worst case pull-down signal path is mapped to an equivalent canonical driver circuit Q5 **206** as shown in Fig. 5. The path subcircuit **207b** is constructed to include one or more of the FETs, Q_{S1} and Q_{Sn} (as shown in Fig. 2A), each included FET representing identified pass circuit elements of the worst case pull-down signal path. The sizes of the

FETs included in the path subcircuit **207** are based on the sizes of the respective pass circuit elements.

In step **305**, a simulation is performed to determine the zero margin ($Vmar_{(0)}$). An exemplary embodiment of the simulation circuit **600** for the $Vmar_{(0)}$ determination is shown in Fig. 6. The $Vmar_{(0)}$ simulation circuit **600** includes the forward inverter portion, i.e., the complementary pair of FETs Q1 and Q2, the NFET Q5, the PFET Q4 and the path subcircuit **207b** constructed in step **304** above. In this exemplary simulation circuit **600**, the FETs Q3 **203** and Q6 **206** shown in Fig. 2 are omitted. However, in an alternative embodiment, the FETs Q3 **203** and Q6 **206** may be included, and turned off.

The $Vmar_{(0)}$ simulation comprises a DC analysis of the simulation circuit **600** with the output Vout **104** initially set to logical low, i.e., set to ground, and determining the voltage level at the input terminal Vin **103**. If the voltage level at the input terminal Vin **103** is below the V_{trip}, and is sufficiently low to overcome the PFET Q4 **204**, the state of the output, Vout **104**, would switch from the low initial setting to a logical high. The zero margin, $Vmar_{(0)}$, is the difference of V_{trip} and the voltage level at Vin **103**, i.e., $Vmar_{(0)} = V_{trip}$ - Vin. Finally, the process ends in step **306**.

If each of the one margin and the zero margin exceed a design guideline margin thresholds, the circuit design is deemed acceptable, and proper operations of the latch is ensured.

As can be appreciated, the DC margin determination method described herein allows an efficient and fast determination of DC margin of a latch without requiring numerous time consuming simulations for each of the possible signal paths to the input of the latch.

Specific program listings for an embodiment of a method of determining the DC margin of a latch in accordance with the principles of the present invention is provided in the Appendix, which appears below.

While the invention has been described with reference to the exemplary embodiments thereof, those skilled in the art will be able to make various modifications to the described embodiments of the invention without departing from the true spirit and scope of the invention. The terms and descriptions used herein are set forth by way of illustration only and are not meant as limitations. In particular, although the method of the present invention has been described by examples, the steps of the method may be performed in a different order than illustrated or simultaneously. Those skilled in the art will

recognize that these and other variations are possible within the spirit and scope of the invention as defined in the following claims and their equivalents.

APPENDIX

```
/* There are 8 FETs in the model and I want to array to index from 1 to 8 */
#define NUM_CKLATCH_FETS 7
#define NUM_PASSCKT_FETS 7
#define ALL_SPICE_FET_PARAMETERS \
fprintf(MYDECK," (m1_w m1_l m2_w m2_l m3_w m3_l m4_w m4_l m5_w m5_l m6_w m6_l)\n") ;
void declare_vth_ckt(MYDECK, lmodel)
FILE* MYDECK ;
int lmodel[NUM_CKLATCH_FETS] ;
{
   fprintf(MYDECK,".subckt vth_ckt 100 %%\"GND\"");
   ALL_SPICE_FET_PARAMETERS;
   fprintf(MYDECK,"M1 3 3 0 0 N%s L=m1_l*1e-6 W=m1_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n", lmodel[1] ?
"L" : "");
   fprintf(MYDECK,"M2 3 3 100 0 P%s L=m2_l*1e-6 W=m2_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n", lmodel[2]
? "L" : "");
   fprintf(MYDECK,".ends vth_ckt\n");
}
#define NOPASSFET_R 1.0e-3
void declare_passfet_ckt(MYDECK, ckt_name, ppass_fets, npass_fets)
FILE* MYDECK ; char* ckt_name ; elem_pr npass_fets[NUM_PASSCKT_FETS],
ppass_fets[NUM_PASSCKT_FETS] ;
{
 int i, left, right, left_portnum, levels = 0;
 /* determine the number of passfet levels. */
 for (i = 1; i < NUM_PASSCKT_FETS; i++) { if ((npass_fets[i] != NULL) || (ppass_fets[i] != NULL)) levels++ ; }
 if (levels == 0) left_portnum = 3;
 else left_portnum = levels + 2 ;
  fprintf(MYDECK,".subckt %s 100 2 %d %%\"GND\" \n", ckt_name, left_portnum) ;
  if (levels == 0)
/* No pass fets, so use a tiny Resistor. */
     fprintf(MYDECK,"R5 3 2 %.2e\n", NOPASSFET_R);
   else
    for (i = 1; i < NUM_PASSCKT_FETS; i++) {
left = i + 2; right = i + 1 ;
     if (npass_fets[i])
 fprintf(MYDECK,"M%d %d 100 %d 0 N%s L=%.2f*1e-6 W=%.2f*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
  2*i -1, left, right, npass_fets[i]->len > CheckLatchLongNLength_ReqERC ? "L" : "",
  npass_fets[i]->len, npass_fets[i]->wid);
     if (ppass_fets[i])
 fprintf(MYDECK,"M%d %d 0 %d 0 P%s L=%.2f*1e-6 W=%.2f*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
  2*i, left, right, ppass_fets[i]->len > CheckLatchLongPLength_ReqERC ? "L" : "" ,
  ppass_fets[i]->len, ppass_fets[i]->wid);
    }
   fprintf(MYDECK,".ends %s\n", ckt_name);
}
void declare_set0_ckt(MYDECK, lmodel)
FILE* MYDECK ; int lmodel[NUM_CKLATCH_FETS] ;
{
   fprintf(MYDECK,".subckt set0_ckt 100 %%\"GND\"") ;
   ALL_SPICE_FET_PARAMETERS
   fprintf(MYDECK,"M1 4 3 0 0 N%s L=m1_l*1e-6 W=m1_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
   lmodel[1] ? "L" : "");
   fprintf(MYDECK,"M2 4 3 100 0 P%s L=m2_l*1e-6 W=m2_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
   lmodel[2] ? "L" : "");
   fprintf(MYDECK,"M4 3 0 100 0 P%s L=m4_l*1e-6 W=m4_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
   lmodel[4] ? "L" : "");
```

```
    fprintf(MYDECK,"X0 100 2 3 %%\"GND\" pulldown_passfet_ckt \n") ;
    fprintf(MYDECK,"M5 2 100 0 0 N%s L=m5_l*1e-6 W=m5_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
    lmodel[5] ? "L" : "");
    fprintf(MYDECK,".ends set0_ckt\n");
}
void declare_set1_ckt(MYDECK, lmodel)
FILE* MYDECK ; int lmodel[NUM_CKLATCH_FETS] ;
{
    fprintf(MYDECK,".subckt set1_ckt 100 %%\"GND\"") ;
    ALL_SPICE_FET_PARAMETERS
    fprintf(MYDECK,"M1 4 3 0 0 N%s L=m1_l*1e-6 W=m1_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
    lmodel[1] ? "L" : "");
    fprintf(MYDECK,"M2 4 3 100 0 P%s L=m2_l*1e-6 W=m2_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
    lmodel[2] ? "L" : "");
    fprintf(MYDECK,"M3 3 100 0 0 N%s L=m3_l*1e-6 W=m3_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
    lmodel[3] ? "L" : "");
    fprintf(MYDECK,"X0 100 2 3 %%\"GND\" pullup_passfet_ckt \n") ;
    fprintf(MYDECK,"M6 2 0 100 0 P%s L=m6_l*1e-6 W=m6_w*1e-6 AD=1.5 AS=1.5 PD=1.5 PS=1.5\n",
    lmodel[6] ? "L" : "");
    fprintf(MYDECK,".ends set1_ckt\n");
}
void add_vth_ckt_to_deck(MYDECK, m_w, m_l)
FILE *MYDECK; double m_w[NUM_CKLATCH_FETS] ; double m_l[NUM_CKLATCH_FETS] ;
{
    fprintf(MYDECK,"X0 100 %%\"GND\" vth_ckt (");
    fprintf(MYDECK,"%.2f %.2f ",m_w[1], m_l[1]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[2], m_l[2]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[3], m_l[3]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[4], m_l[4]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[5], m_l[5]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[6], m_l[6]) ;
    fprintf(MYDECK,")\n") ;
}
void add_set0_ckt_to_deck(MYDECK, m_w, m_l)
FILE *MYDECK; double m_w[NUM_CKLATCH_FETS] ; double m_l[NUM_CKLATCH_FETS] ;
{
    fprintf(MYDECK,"X1 100 %%\"GND\" set0_ckt (");
    fprintf(MYDECK,"%.2f %.2f ",m_w[1], m_l[1]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[2], m_l[2]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[3], m_l[3]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[4], m_l[4]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[5], m_l[5]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[6], m_l[6]) ;
    fprintf(MYDECK,")\n") ;
}
void add_set1_ckt_to_deck(MYDECK, m_w, m_l)
FILE *MYDECK; double m_w[NUM_CKLATCH_FETS] ; double m_l[NUM_CKLATCH_FETS] ;
{
    fprintf(MYDECK,"X2 100 %%\"GND\" set1_ckt (");
    fprintf(MYDECK,"%.2f %.2f ",m_w[1], m_l[1]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[2], m_l[2]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[3], m_l[3]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[4], m_l[4]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[5], m_l[5]) ;
    fprintf(MYDECK,"%.2f %.2f ",m_w[6], m_l[6]) ;
    fprintf(MYDECK,")\n") ;
}
/* Return TRUE if this latch has tristate feedback. */
```

```
1  int latch_has_tristate_feedback(node) node_pr node ;
2  {
3   node_pr output, inv_output, tsinv_node ;
4   elem_pr elem ;
5   if (!NIsLatch(node)) return FALSE ;
6   tsinv_node = NTriStateInvOf(node) ;
7   for_gate_elems(elem, node) {
8    output = get_output(elem) ;
9    if (!output) continue ;
10   inv_output = NStatInvOf(output) ;
11   if (NSame(inv_output, node) && NSame(output, tsinv_node)) { return TRUE ; }
12  } end_gate_elems
13  return FALSE ;
14 }
15 /* Given a latch node return the tristate feedback node. */
16 node_pr get_tristate_feedback(node)
17 node_pr node ;
18 {
19  elem_pr elem ;
20  node_pr output, inv_output, tsinv_node ;
21  if (!NIsLatch(node)) return NULL ;
22  tsinv_node = NTriStateInvOf(node) ;
23  for_gate_elems(elem, node) {
24   output = get_output(elem) ;
25   if (!output) continue ;
26   inv_output = NStatInvOf(output) ;
27   if (NSame(inv_output, node) && NSame(output, tsinv_node)) { return output ; }
28  } end_gate_elems
29  return NULL ;
30 }
31 /* Find the FET widths and lengths for this node.  Return a non zero if there is a problem running. */
32 #define ABSENT_FEEDBACK_FET_L_MULT 20.0
33 int find_checklatch_latchfets(node, m_w, m_l)
34 node_pr node ; double *m_w, *m_l ;
35 {
36  elem_pr elem, nelem;
37  node_pr output, inv_output, inv_node, tsinv_node, gate, inv_gate, ngate, inv_ngate, ts_feedback, other_side;
38  int has_tsfeedback, has_single_feedback, has_inv_feedback ;
39  inv_node = NStatInvOf(node) ;
40  tsinv_node = NTriStateInvOf(node) ;
41  has_tsfeedback = latch_has_tristate_feedback(node) ;
42  has_single_feedback = (!inv_node && !tsinv_node) ? TRUE : FALSE ;
43  has_inv_feedback = (!has_tsfeedback && !has_single_feedback) ? TRUE : FALSE ;
44  /* Find the forward inverter FETs. */
45  if (has_single_feedback) {
46   for_gate_elems(elem, node) {
47    output = get_output(elem) ;
48    if (!output) continue ;
49    inv_output = NStatInvOf(output) ;
50    if (NSame(inv_output, node)) {
51     if (ETypeIsP(elem)) { m_w[2] = elem->wid ; m_l[2] = elem->len ; }
52     if (ETypeIsN(elem)) { m_w[1] = elem->wid ; m_l[1] = elem->len ; }
53    }
54   } end_gate_elems
55  }
56  if (has_inv_feedback || has_tsfeedback) {
57   for_gate_elems(elem, node) {
58    output = get_output(elem) ;
```

```
    if (!output) continue ;
    inv_output = NStatInvOf(output) ;
    if ( (NSame(inv_output, node) && NSame(output, inv_node)) ||
        (NSame(inv_output, node) && NSame(output, tsinv_node)) ) {
     if (ETypeIsP(elem)) { m_w[2] = elem->wid ; m_l[2] = elem->len ; }
     if (ETypeIsN(elem)) { m_w[1] = elem->wid ; m_l[1] = elem->len ; }
    }
   } end_gate_elems
  }
  if (m_w[1] == 0 || m_w[2] == 0) return 1 ;
  /* Find the feedback inverter FETs. */
  if (has_single_feedback) {
   for_chan_fets(elem, node) {
    gate = EGate(elem) ;
    inv_gate = NStatInvOf(gate) ;
    if (NSame(node, inv_gate)) {
     if (ETypeIsP(elem)) { m_w[4] = elem->wid ; m_l[4] = elem->len ; }
     if (ETypeIsN(elem)) { m_w[3] = elem->wid ; m_l[3] = elem->len ; }
    }
   } end_chan_fets
  }
  if (has_tsfeedback) {
   ts_feedback = get_tristate_feedback(node) ;
   for_chan_fets(elem, node) {
    gate = EGate(elem) ;
    inv_gate = NStatInvOf(gate) ;
    if (NSame(node, inv_gate) && NSame(ts_feedback, gate) && ETypeIsP(elem)) {
     m_w[4] = elem->wid ; m_l[4] = elem->len ;
    }
    if (ETypeIsN(elem)) {
/* Type I tristate inverter. */
if (NSame(node, inv_gate) && NSame(ts_feedback, gate)) { m_w[3] = elem->wid ; m_l[3] = elem->len ; }
/* Type II tristate inverter. */
else {
 other_side = EOtherChan(elem, node) ;
 nested_for_chan_elems(nelem, other_side) {
  if (ETypeIsN(nelem)) {
       ngate = EGate(nelem) ;
       inv_ngate = NStatInvOf(ngate) ;
   if (NSame(node, inv_ngate) && NSame(ts_feedback, ngate)) { m_w[3] = nelem->wid ; m_l[3] = nelem->len ; }
  }
 } nested_end_chan_elems
}
    }
   } end_chan_fets
  }
  if (has_inv_feedback) {
   for_chan_fets(elem, node) {
    gate = EGate(elem) ;
    inv_gate = NStatInvOf(gate) ;
    if (NSame(node, inv_gate) && NSame(inv_node, gate)) {
     if (ETypeIsP(elem)) { m_w[4] = elem->wid ; m_l[4] = elem->len ; }
     if (ETypeIsN(elem)) { m_w[3] = elem->wid ; m_l[3] = elem->len ; }
    }
   } end_chan_fets
  }
  /* Some latches have missing Feedback FETs */
  if (m_w[3] == 0  && m_w[4] == 0) return 1 ;
```

```
 if (m_w[3] == 0) { m_w[3] = MinGateZ_ReqERC ; m_l[3] = ABSENT_FEEDBACK_FET_L_MULT *
DeviceL_ReqERC ; }
 if (m_w[4] == 0) { m_w[4] = MinGateZ_ReqERC ; m_l[4] = ABSENT_FEEDBACK_FET_L_MULT *
DeviceL_ReqERC ; }
 return 0 ;
}
/* For a given tree, write into the deck the element names.  This is a recursive routine. */
void add_fet_tree_comments_to_deck(DECK, dlnode_ptr)
FILE* DECK ; dlnodeptr dlnode_ptr;
{
 char message[MAXSTRING] ;
 elem_pr this_elem, parallel_elem ;
 /* Go across */
 if (dlnode_ptr->next) { add_fet_tree_comments_to_deck(DECK, dlnode_ptr->next); }
 /* Go down if FETs exist there */
 if (dlnode_ptr->down) { add_fet_tree_comments_to_deck(DECK, dlnode_ptr->down); }
 this_elem = dlnode_ptr->elem ;
 sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f",
  this_elem->name, this_elem->gate->name, this_elem->source->name, this_elem->drain->name,
this_elem->wid) ;
 add_comment_to_deck(DECK, message) ;
 /* A channel parallel element will not show up in the tree, but will be used in effective_w_calc_from_tree */
 parallel_elem = this_elem->chan_parallel ;
 if (parallel_elem && !ESame(parallel_elem, this_elem)) {
  sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f",
   parallel_elem->name, parallel_elem->gate->name, parallel_elem->source->name, parallel_elem->drain->name,
parallel_elem->wid) ;
  add_comment_to_deck(DECK, message) ;
 }
}
/* Annotate the deck with the FETs used to in build_generic_tree.  Also record in the deck all the pass fets
included in the pullup and pulldown models.  */
void add_checklatch_comments_to_deck(MYDECK, pulldown_ppass_fets, pulldown_npass_fets,
pullup_ppass_fets,
pullup_npass_fets, ndriver_node, pdriver_node)
elem_pr pulldown_npass_fets[NUM_PASSCKT_FETS], pulldown_ppass_fets[NUM_PASSCKT_FETS] ;
elem_pr pullup_npass_fets[NUM_PASSCKT_FETS], pullup_ppass_fets[NUM_PASSCKT_FETS] ;
node_pr pdriver_node, ndriver_node ; FILE* MYDECK ;
{
 elem_pr elem, this_elem;
 dlnodeptr nfet_tree_ptr,pfet_tree_ptr;
 double nfet_max_l_over_w,nfet_min_l_over_w,nfet_in_parallel;
 double pfet_max_l_over_w,pfet_min_l_over_w,pfet_in_parallel;
 int node_type_flag, i;
 char message[MAXSTRING] ;
 add_comment_to_deck(MYDECK, " Pullup path passfets info: ") ;
 if (pullup_npass_fets[1] && pullup_ppass_fets[1])
  add_comment_to_deck(MYDECK, " Latch pullup passfet structure: COMPLIMENTARY ") ;
 else if (pullup_npass_fets[1])
  add_comment_to_deck(MYDECK, " Latch pullup passfet structure: NFET ") ;
 else if (pullup_ppass_fets[1])
  add_comment_to_deck(MYDECK, " Latch pullup passfet structure: PFET ") ;
 else
  add_comment_to_deck(MYDECK, " Latch pullup passfet structure: NONE ") ;
 for (i = 1; i < NUM_PASSCKT_FETS; i++) {
  if (pullup_npass_fets[i]) {
   this_elem = pullup_npass_fets[i] ;
   sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f",
```

```
      this_elem->name, this_elem->gate->name, this_elem->source->name, this_elem->drain->name,
this_elem->wid) ;
     add_comment_to_deck(MYDECK, message) ;
    }
    if (pullup_ppass_fets[i]) {
     this_elem = pullup_ppass_fets[i] ;
     sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f",
      this_elem->name, this_elem->gate->name, this_elem->source->name, this_elem->drain->name,
this_elem->wid) ;
     add_comment_to_deck(MYDECK, message) ;
    }
   }
   add_comment_to_deck(MYDECK, " Pulldown path passfets info: ") ;
   if (pulldown_npass_fets[1] && pulldown_ppass_fets[1])
    add_comment_to_deck(MYDECK, " Latch pulldown passfet structure: COMPLIMENTARY ") ;
   else if (pulldown_npass_fets[1])
    add_comment_to_deck(MYDECK, " Latch pulldown passfet structure: NFET ") ;
   else if (pulldown_ppass_fets[1])
    add_comment_to_deck(MYDECK, " Latch pulldown passfet structure: PFET ") ;
   else
    add_comment_to_deck(MYDECK, " Latch pulldown passfet structure: NONE ") ;
   for (i = 1; i < NUM_PASSCKT_FETS; i++) {
    if (pulldown_npass_fets[i]) {
     this_elem = pulldown_npass_fets[i] ;
     sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f", this_elem->name, this_elem->gate->name,
this_elem->source->name, this_elem->drain->name, this_elem->wid) ;
     add_comment_to_deck(MYDECK, message) ;
    }
    if (pulldown_ppass_fets[i]) {
     this_elem = pulldown_ppass_fets[i] ;
     sprintf(message, "%s gate: %s source: %s drain: %s width: %.4f", this_elem->name, this_elem->gate->name,
this_elem->source->name, this_elem->drain->name, this_elem->wid) ;
     add_comment_to_deck(MYDECK, message) ;
    }
   }
   /* Include as comments all the FETs that make up the generic trees. */
   nfet_tree_ptr = build_generic_tree(ndriver_node,NTYPE,0,&node_type_flag);
   pfet_tree_ptr = build_generic_tree(pdriver_node,PTYPE,0,&node_type_flag);
   if (nfet_tree_ptr) {
    add_comment_to_deck(MYDECK, "NFET driver values:") ;
    add_fet_tree_comments_to_deck(MYDECK,nfet_tree_ptr);
    free_fet_tree(nfet_tree_ptr,nfet_tree_ptr);
   }
   else { add_comment_to_deck(MYDECK, "Using Ported NFET driver value") ; }
   if (pfet_tree_ptr) {
    add_comment_to_deck(MYDECK, "PFET driver values:") ;
    add_fet_tree_comments_to_deck(MYDECK,pfet_tree_ptr);
    free_fet_tree(pfet_tree_ptr,pfet_tree_ptr);
   }
   else { add_comment_to_deck(MYDECK, "Using Ported PFET driver value") ; }
  }
  /* Given two nodes, find one P and one N pass fet between these two nodes if there are any.  If there are more
  than one pair of N and P, then this routine returns the first two that it finds. */
  void find_passfets_between_nodes(node, other_node, n_elem, p_elem)
  node_pr node, other_node ; elem_pr *n_elem, *p_elem ;
  {
   elem_pr elem ;
   node_pr other_node2 ;
```

```
1   *n_elem = *p_elem = NULL ;
2   for_chan_elems(elem, other_node) {
3     if (EIsPassFet(elem)) {
4       other_node2 = EOtherChan(elem,other_node) ;
5       if (NSame(other_node2,node)) {
6         if (ETypeIsN(elem)) *n_elem = elem ;
7   if (ETypeIsP(elem)) *p_elem = elem ;
8       }
9     }
10   end_chan_elems }
11  }
12  /* Given a latch node, find the node that in turn provide the worst case pullup toVDD or the worst case
13  pulldown to GND. Assume that this node is a pass fet output. This routine calls itself until it is on a node which
14  is a passfet input and not a passfet output. It keeps track of the current worst case driver and updates a higher
15  level record of the worst case driver's characteristicsas it goes along.  Search only in the direction of signal flow.
16  */
17  void find_checklatch_driver_r(node, wc_node, type, wc_width, wc_loverw, ppass_fets, npass_fets, level,
18  wc_ppass_fets, wc_npass_fets, wc_levels)
19  node_pr node ; node_pr* wc_node ; int type, level ; int *wc_levels ;
20  elem_pr *ppass_fets, *npass_fets ; elem_pr *wc_ppass_fets, *wc_npass_fets ; double *wc_width,
21  *wc_loverw ;
22  {
23   elem_pr elem ; node_pr other_node ; dlnodeptr fet_tree_ptr,pfet_tree_ptr;
24   double fet_max_l_over_w,fet_min_l_over_w,fet_in_parallel;
25   int is_npass, is_ppass, i, node_type_flag, nfet_vt_drop, pfet_vt_drop ;
26   double node_width, total_loverw, stage_loverw, nscale, pscale ; elem_pr n_elem, p_elem ; double nfet_loverw,
27  pfet_loverw ;
28   for_chan_fets(elem, node) {
29     other_node = EOtherChan(elem,node) ;
30     if (NIsPassGateIn(other_node) && !NIsMarked(other_node) && NSame(other_node, EInputChan(elem))) {
31       push_node_set_mark(other_node) ;
32       /* The calculation of cumulative l_over_w depends on the type of pass fet. */
33       find_passfets_between_nodes(node, other_node, &n_elem, &p_elem) ;
34       if (level < NUM_PASSCKT_FETS - 1) {
35         if (n_elem) npass_fets[level + 1] = n_elem ;
36         if (p_elem) ppass_fets[level + 1] = p_elem ;
37       }
38       find_checklatch_driver_r(other_node, wc_node, type, wc_width, wc_loverw, ppass_fets, npass_fets,
39  level+1, wc_ppass_fets, wc_npass_fets, wc_levels) ;
40     }
41   } end_chan_fets
42   if (NIsPassGateOut(node)) return ;
43   fet_tree_ptr = build_generic_tree(node,type,0,&node_type_flag);
44   if (fet_tree_ptr) {
45     combine_shared_diffusions(fet_tree_ptr,fet_tree_ptr);
46     effective_w_calc_from_tree(fet_tree_ptr, &fet_max_l_over_w, &fet_min_l_over_w, &fet_in_parallel);
47     node_width = DeviceL_ReqERC / fet_max_l_over_w ;
48     free_fet_tree(fet_tree_ptr,fet_tree_ptr);
49   }
50   else node_width = (type == NTYPE) ? CheckLatchDefaultNWidth_ReqERC :
51  CheckLatchDefaultPWidth_ReqERC ;
52
53   /* The total L/W for this path depends on a weighted sum of the L/W for the pass FETs along the path.  It also
54  depends on whether this is a pullup or pulldown path.  */
55   pfet_vt_drop = nfet_vt_drop = FALSE ;
56   if (type == PTYPE) total_loverw = (DeviceL_ReqERC/node_width) * PMobility_ReqERC *
57  CheckLatchPFETPassingVDD_ReqERC ;
58   else total_loverw = (DeviceL_ReqERC/node_width) * CheckLatchNFETPassingGND_ReqERC ;
```

```
1    for (i = NUM_PASSCKT_FETS - 1; i > 0; i--) {
2       is_npass = (npass_fets[i] != NULL) ? TRUE : FALSE ;
3       is_ppass = (ppass_fets[i] != NULL) ? TRUE : FALSE ;
4       if (!is_ppass && !is_npass) continue ;
5       if (is_ppass) { pfet_loverw = PMobility_ReqERC * ((ppass_fets[i])->len / (ppass_fets[i])->wid) ; }
6       if (is_npass) { nfet_loverw = (npass_fets[i])->len / (npass_fets[i])->wid  ; }
7       /* The L/W for this passfet stage depends on whether this is a pullup
8             or pulldown path and on whether there has been a Vt drop along this path. */
9       if (type == NTYPE) {
10   /* pulldown path */
11   pscale = pfet_vt_drop ? CheckLatchPFETPassingGND_Vt_ReqERC : CheckLatchPFETPassingGND_ReqERC ;
12   nscale = nfet_vt_drop ? CheckLatchNFETPassingGND_Vt_ReqERC : CheckLatchNFETPassingGND_ReqERC ;
13      }
14      else {
15   /* pullup path */
16   pscale = pfet_vt_drop ? CheckLatchPFETPassingVDD_Vt_ReqERC : CheckLatchPFETPassingVDD_ReqERC ;
17   nscale = nfet_vt_drop ? CheckLatchNFETPassingVDD_Vt_ReqERC : CheckLatchNFETPassingVDD_ReqERC ;
18      }
19      pfet_loverw *= pscale ;
20      nfet_loverw *= nscale ;
21
22      if (is_npass && is_ppass) { stage_loverw = (nfet_loverw * pfet_loverw) / (nfet_loverw + pfet_loverw) ; }
23      else if (is_npass && !is_ppass) { stage_loverw = nfet_loverw ; }
24      else { stage_loverw = pfet_loverw ; }
25      total_loverw += stage_loverw ;
26
27      /* For the next set of pass fets, remember if there has been a Vt drop. */
28      if ( is_ppass && !is_npass) pfet_vt_drop |= TRUE ;
29      if (!is_ppass &&  is_npass) nfet_vt_drop |= TRUE ;
30    }
31    /* If this current node is worse than the current worst offender, record the information. */
32    if (total_loverw > *wc_loverw) {
33      *wc_width = node_width ;
34      *wc_node = node ;
35      *wc_loverw = total_loverw ;
36      for (i = 0; i < NUM_PASSCKT_FETS; i++)  {
37        wc_npass_fets[i] = npass_fets[i] ;
38        wc_ppass_fets[i] = ppass_fets[i] ;
39      }
40      *wc_levels = level ;
41    }
42    /* Remove passfet information that was recorded for this level. */
43    if (level < NUM_PASSCKT_FETS) {
44      npass_fets[level] = ppass_fets[level] = NULL ;
45    }
46  }
47  /* Given a node that is the input to a latch, traverse over any other pass FETs and determine the weakest
48  equivalent width FET that either pulls up or down the latch input. What is considered "weakest" has changed
49  over time to try to match what checklatch classic calls the "weakest" path. This routine also keeps track of all the
50  passfets that are encountered on the way to the worst case driver. It also keeps track of the number of passfet
51  levels that were found in getting to the worst case driver.*/
52  void find_checklatch_driver(node, wc_node, type, wc_width, wc_ppass_fets, wc_npass_fets, wc_levels)
53  node_pr node, *wc_node ; int type, *wc_levels ; double* wc_width ; elem_pr *wc_ppass_fets,
54  *wc_npass_fets ;
55  {
56    elem_pr npass_fets[NUM_PASSCKT_FETS], ppass_fets[NUM_PASSCKT_FETS] ;
57    double wc_loverw = 0.0 ; int i ;
58    for (i = 0; i < NUM_PASSCKT_FETS; i++)  { npass_fets[i] = ppass_fets[i] = NULL ; }
```

```
 1    push_node_set_mark(node) ;
 2    find_checklatch_driver_r(node, wc_node, type, wc_width, &wc_loverw, ppass_fets, npass_fets, 0,
 3   wc_ppass_fets, wc_npass_fets, wc_levels) ;
 4    clear_node_marks() ;
 5   }
 6   /* Return TRUE if this node can be determined to be the feedback node of a latch structure. */
 7   int NIsFeedBack(node)
 8   node_pr node ;
 9   {
10    elem_pr elem ; int n_feedback = FALSE, p_feedback = FALSE ;
11    if (!NIsLatch(node)) return FALSE ;
12    for_gate_elems(elem, node) {
13     if(EIsFeedback(elem))
14      if (ETypeIsN(elem)) n_feedback = TRUE ;
15      if (ETypeIsP(elem)) p_feedback = TRUE ;
16    } end_gate_elems
17    if (n_feedback && p_feedback) return TRUE ;
18    else return FALSE ;
19   }
20   /* Write one deck with 3 circuits in it for each latch node.  This is the top level routine for creating the checklatch
21   decks. */
22   check_latch_node(node)
23   node_pr node ;
24   {
25    FILE *MYDECK ; elem_pr pullup_npass_fets[NUM_PASSCKT_FETS],
26   pullup_ppass_fets[NUM_PASSCKT_FETS] ;
27    elem_pr pulldown_npass_fets[NUM_PASSCKT_FETS], pulldown_ppass_fets[NUM_PASSCKT_FETS] ;
28    elem_pr elem; node_pr pdriver_node = NULL, ndriver_node = NULL ;
29    double m_w[NUM_CKLATCH_FETS], m_l[NUM_CKLATCH_FETS] ; int i, lmodel[NUM_CKLATCH_FETS] ;
30    int pullup_levels = 0, pulldown_levels = 0 ; char message[MAXSTRING];
31
32    /* Skip this node if it's the output of the forward inverter. */
33    if (NIsFeedBack(node)) return ;
34
35    for (i = 0; i < NUM_CKLATCH_FETS; i++) { m_w[i] = 0.0; m_l[i] = DeviceL_ReqERC ; }
36    for (i = 0; i < NUM_PASSCKT_FETS; i++)  { pullup_npass_fets[i] = pullup_ppass_fets[i] =
37   pulldown_npass_fets[i] = pulldown_ppass_fets[i] = NULL ; }
38
39    /* determine the fet widths and lengths for this node */
40    if (find_checklatch_latchfets(node, &m_w[0], &m_l[0])) return ;
41
42    find_checklatch_driver(node, &pdriver_node, PTYPE, &m_w[6], pullup_ppass_fets, pullup_npass_fets,
43   &pullup_levels) ;
44    find_checklatch_driver(node, &ndriver_node, NTYPE, &m_w[5], pulldown_ppass_fets, pulldown_npass_fets,
45   &pulldown_levels) ;
46    if (!pdriver_node || !ndriver_node) return ;
47
48    /* The NFETS have odd transistor numbers, the PFETs have even numbers. */
49    for (i = 1; i < NUM_CKLATCH_FETS; i+=2) lmodel[i] = m_l[i] > CheckLatchLongNLength_ReqERC ? TRUE :
50   FALSE ;
51    for (i = 2; i < NUM_CKLATCH_FETS; i+=2) lmodel[i] = m_l[i] > CheckLatchLongPLength_ReqERC ? TRUE :
52   FALSE ;
53
54    MYDECK = start_deck(CHECKLATCH_SIM, node);
55    declare_passfet_ckt(MYDECK, "pulldown_passfet_ckt", pulldown_ppass_fets, pulldown_npass_fets ) ;
56    declare_passfet_ckt(MYDECK, "pullup_passfet_ckt", pullup_ppass_fets, pullup_npass_fets ) ;
57    declare_vth_ckt(MYDECK, lmodel) ;
58    declare_set0_ckt(MYDECK, lmodel) ;
```

```
1   declare_set1_ckt(MYDECK, lmodel) ;
2   add_checklatch_comments_to_deck(MYDECK, pulldown_ppass_fets, pulldown_npass_fets,
3  pullup_ppass_fets,
4     pullup_npass_fets, ndriver_node, pdriver_node) ;
5
6   /* Place a message about the total number of levels */
7   if ((pullup_levels >= NUM_PASSCKT_FETS) || (pulldown_levels >= NUM_PASSCKT_FETS)) {
8     sprintf(message,"Model Error: Total pullup passfets %d Total pulldown passfets %d Model limit %d",
9       pullup_levels, pulldown_levels, NUM_PASSCKT_FETS -1) ;
10    add_comment_to_deck(MYDECK, message) ;
11  }
12
13  add_ckt_header_to_deck(MYDECK, "ckt1", CHECKLATCH_QUERY) ;
14  fprintf(MYDECK,"V1 100 0 dc=%.4e \n", CheckLatchSupplyVoltage_ReqERC);
15  add_vth_ckt_to_deck(MYDECK, m_w, m_l) ;
16  add_set0_ckt_to_deck(MYDECK, m_w, m_l) ;
17  add_set1_ckt_to_deck(MYDECK, m_w, m_l) ;
18  end_deck(MYDECK);
19 }
20
```

CLAIMS

What is claimed is:

1. A method of determining a DC margin of a latch, comprising:

performing a first simulation using a first simulation circuit to determine a trip voltage of a forward inverter of said latch;

performing a second simulation using a second simulation circuit to determine a one margin of said latch, said second simulation circuit comprising a worst case pull-up signal path; and

performing a third simulation using a third simulation circuit to determine a zero margin of said latch, said third simulation circuit comprising a worst case pull-down signal path .

2. The method of determining a DC margin of a latch in accordance with claim 1, further comprising:

determining said worst case pull-up signal path analytically by comparing a cumulative weighted resistance of at least one pull-up signal path to said latch.

3. The method of determining a DC margin of a latch in accordance with claim 2, wherein said step of determining said worst case pull-up signal path comprises:

identifying one or more pass circuit elements along each of possible pull-up signal paths;

determining a resistance of each of said identified one or more pass circuit elements based on respective sizes of said identified one or more pass circuit elements;

applying to said resistance, of each of said identified one or more pass circuit elements, a weight based on topology configuration of a corresponding one of said identified one or more pass circuit elements to produce one or more weighted resistance;

summing said one or more weighted resistance to produce said cumulative weighted resistance of each of said possible pull-up signal paths; and

determining said worst case pull-up signal path having a highest cumulative weighted resistance among said possible pull-up signal paths.

4. The method of determining a DC margin of a latch in accordance with claim 1, further comprising:

determining said worst case pull-down signal path analytically by comparing a cumulative weighted resistance of at least one pull-down signal path to said latch.

5. The method of determining a DC margin of a latch in accordance with claim 4, wherein said step of determining said worst case pull-down signal path comprises:

identifying one or more pass circuit elements along each of possible pull-down signal paths;

determining a resistance of each of said identified one or more pass circuit elements based on respective sizes of said identified one or more pass circuit elements;

applying to said resistance, of each of said identified one or more pass circuit elements, a weight based on topology configuration of a corresponding one of said identified one or more pass circuit elements to produce one or more weighted resistance;

summing said one or more weighted resistance to produce said cumulative weighted resistance of each of said possible pull-down signal paths; and

determining said worst case pull-down signal path having a highest cumulative weighted resistance among said possible pull-down signal paths.

6. The method of determining a DC margin of a latch in accordance with claim 1, wherein said step of performing said second simulation comprises:

setting an initial value of an output, of a portion of said second simulation circuit representing said latch, to a logical high;

applying a logical high pull-up input signal to an input, of said portion of said second simulation circuit representing said latch, through said worst case pull-up signal path; and

determining a voltage level at said input.

7. The method of determining a DC margin of a latch in accordance with claim 1, wherein said step of performing said third simulation comprises:

setting an initial value of an output, of a portion of said third simulation circuit representing said latch, to a logical low;

applying a logical low pull-down input signal to an input, of said portion of said third simulation circuit representing said latch, through said worst case pull-down signal path; and

determining a voltage level at said input.

8. A computer readable storage medium having stored thereon computer program for implementing a method of determining a DC margin of a latch, said computer program comprising a set of instructions for:

performing a first simulation using a first simulation circuit to determine a trip voltage of a forward inverter of said latch;

performing a second simulation using a second simulation circuit to determine a one margin of said latch, said second simulation circuit comprising a worst case pull-up signal path; and

performing a third simulation using a third simulation circuit to determine a zero margin of said latch, said third simulation circuit comprising a worst case pull-down signal path .

9. The computer readable storage medium according to claim 8, wherein said computer program further comprising one or more instructions for:

determining said worst case pull-up signal path analytically by comparing a cumulative weighted resistance of at least one pull-up signal path to said latch.

10. The computer readable storage medium according to claim 9, wherein said computer program further comprising one or more instructions for:

identifying one or more pass circuit elements along each of possible pull-up signal paths;

determining a resistance of each of said identified one or more pass circuit elements based on respective sizes of said identified one or more pass circuit elements;

applying to said resistance, of each of said identified one or more pass circuit elements, a weight based on topology configuration of a corresponding one of said identified one or more pass circuit elements to produce one or more weighted resistance;

summing said one or more weighted resistance to produce said cumulative weighted resistance of each of said possible pull-up signal paths; and

determining said worst case pull-up signal path having a highest cumulative weighted resistance among said possible pull-up signal paths.

11. The computer readable storage medium according to claim 8, wherein said computer program further comprising one or more instructions for:

determining said worst case pull-down signal path analytically by comparing a cumulative weighted resistance of at least one pull-down signal path to said latch.

12. The computer readable storage medium according to claim 11, wherein said computer program further comprising one or more instructions for:

identifying one or more pass circuit elements along each of possible pull-down signal paths;

determining a resistance of each of said identified one or more pass circuit elements based on respective sizes of said identified one or more pass circuit elements;

applying to said resistance, of each of said identified one or more pass circuit elements, a weight based on topology configuration of a corresponding one of said identified one or more pass circuit elements to produce one or more weighted resistance;

summing said one or more weighted resistance to produce said cumulative weighted resistance of each of said possible pull-down signal paths; and

determining said worst case pull-down signal path having a highest cumulative weighted resistance among said possible pull-down signal paths.

13. The computer readable storage medium according to claim 8, wherein said one or more instructions for performing said second simulation comprises one or more instructions for:

setting an initial value of an output, of a portion of said second simulation circuit representing said latch, to a logical high;

applying a logical high pull-up input signal to an input, of said portion of said second simulation circuit representing said latch, through said worst case pull-up signal path; and

determining a voltage level at said input.

14. The computer readable storage medium according to claim 8, wherein said one or more instructions for performing said third simulation comprises one or more instructions for:

setting an initial value of an output, of a portion of said second simulation circuit representing said latch, to a logical low;

applying a logical low pull-down input signal to an input, of said portion of said second simulation circuit representing said latch, through said worst case pull-down signal path; and

determining a voltage level at said input.

15. A simulation circuit for determining a DC margin of a latch, comprising:

a latch portion representing said latch being simulated, said latch portion comprising a forward inverter and a feedback inverter, an input of said forward inverter being operably connected to an input of said latch portion, and an input of said feedback inverter being operably connected to an output of said latch portion;

a driver portion representing a driver circuit element capable of supplying an input signal to said latch being simulated; and

a pass path subcircuit configured to receive a drive signal from said driver portion, and configured to supply said drive signal to said input of said latch portion, said pass path subcircuit representing one or more pass circuit elements along a worst case signal path between said driver circuit element and said latch being simulated.

16. The simulation circuit according to claim 15, wherein each of said forward inverter and said feedback inverter comprises:

a complementary pair of field effect transistors.

17. The simulation circuit according to claim 15, wherein said pass path subcircuit comprises:

one or more field effect transistors.

18. The simulation circuit according to claim 17, wherein:

at least two of said one or more field effect transistors are arranged as a complementary pair.

19. The simulation circuit according to claim 17, wherein:

each of said one or more field effect transistors representing corresponding one of said one or more pass circuit elements along a worst case signal path between said driver circuit element and said latch being simulated; and

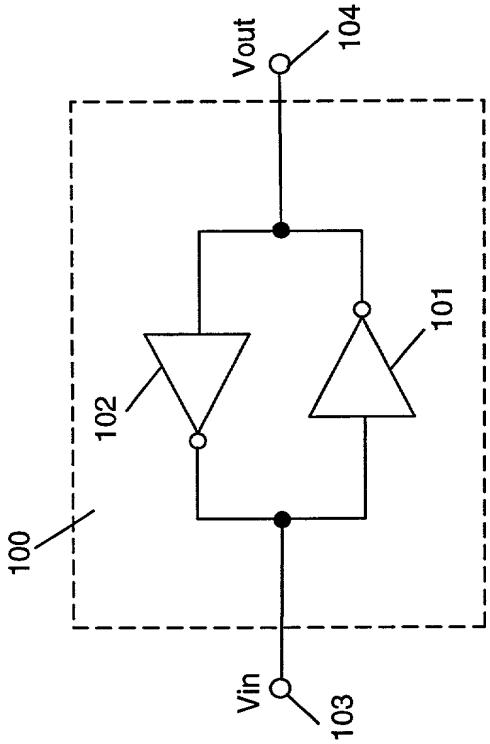
wherein sizes of said one or more field effect transistors are chosen based on respective resistance of corresponding ones of said one or more pass circuit element.

20. The simulation circuit according to claim 19, wherein:

each of said resistance is weighted based on a topology configuration of corresponding one of said one or more pass circuit elements.

ABSTRACT OF THE DISCLOSURE

DC margin of a latch of a circuit under design is determined by performing three simulations. A simulation is performed to find the trip voltage of the forwarding inverter of the latch. A second simulation is performed to find the one margin of the latch. Lastly, a third simulation is performed to find the zero margin of the latch. During each of the simulations to find the one margin and the zero margin, the worst case input signal path from the various driver circuit elements and signal paths within the circuit under design is determined analytically by accumulating weighted resistance of each of the circuit elements along the signal paths. The weights assigned to the circuit elements are empirically determined based on the topology configuration of each of the circuit elements, e.g., the type circuit element, the signal being passed through the circuit element and whether a threshold voltage drop occurs between the drive circuit element and the pass circuit element.



[PRIOR ART]

Fig. 1

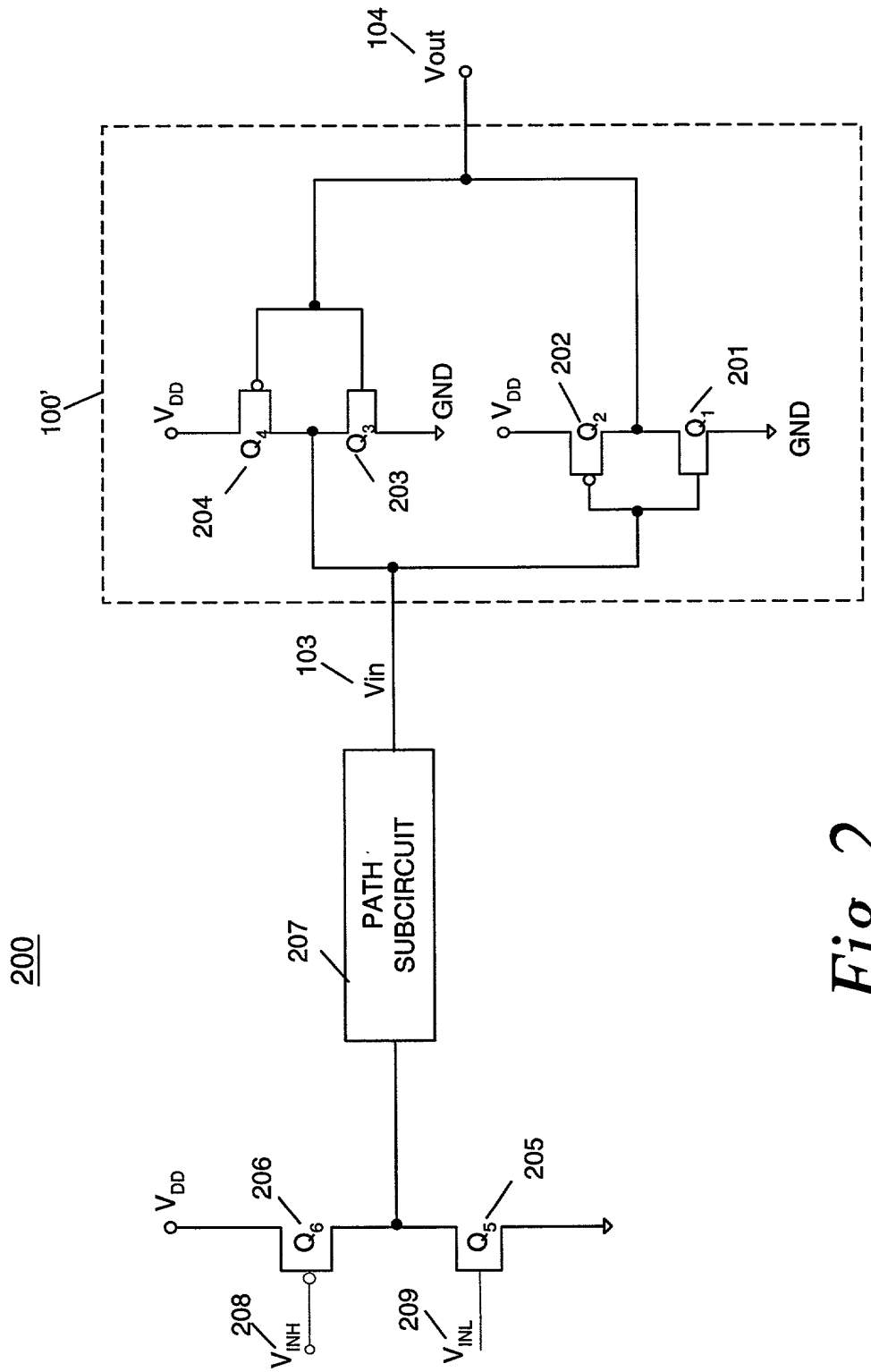

200
100'
104
Vout
V_{DD}
Q_4
Q_3
GND
204
203
V_{DD}
202
Q_2
Q_1
201
GND
103
Vin
PATH
SUBCIRCUIT
207
V_{DD}
206
Q_6
Q_5
205
208
V_{INH}
209
V_{INL}

Fig. 2

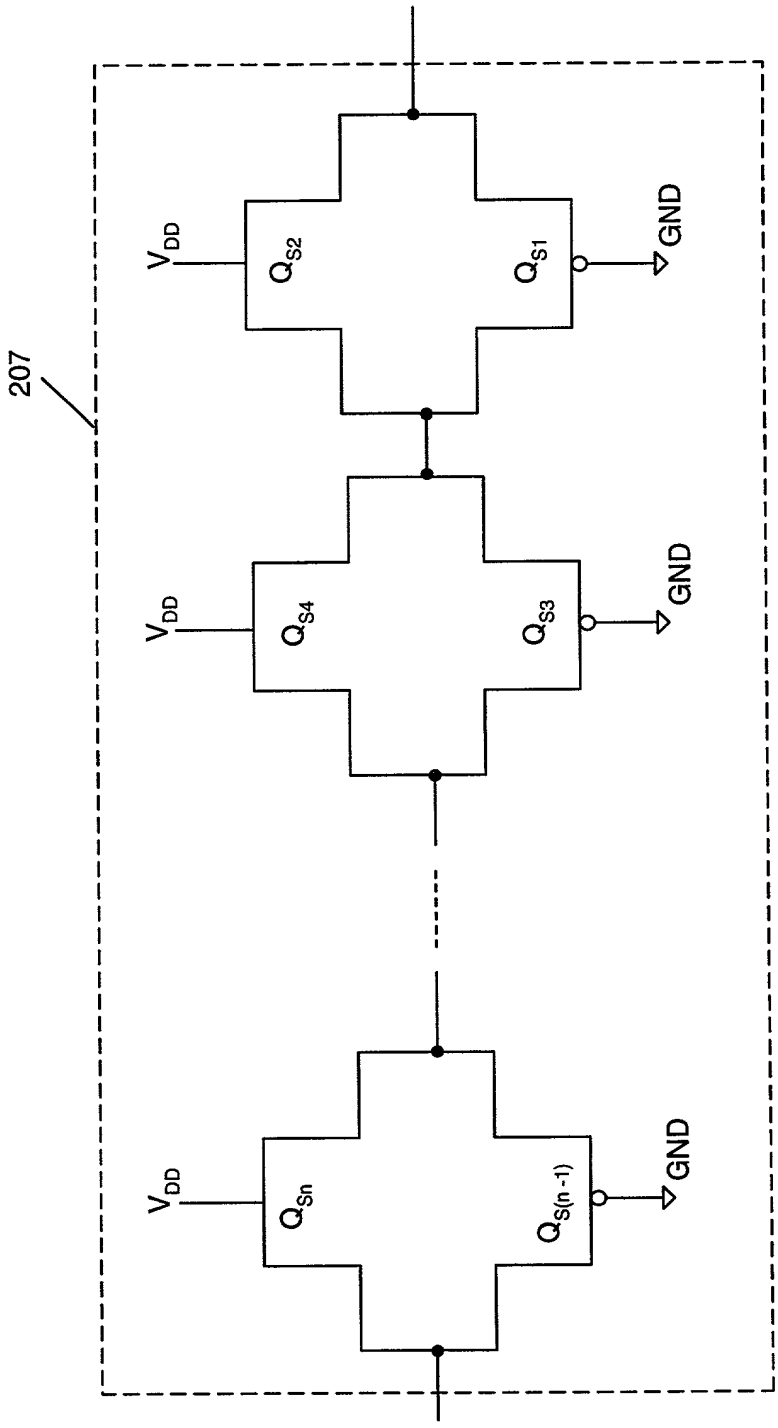


Fig. 2A

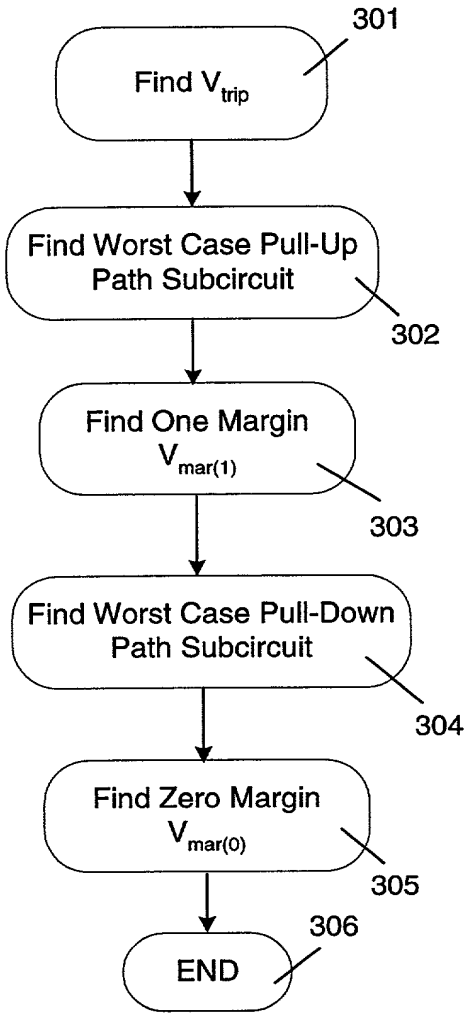

301
Find V_{trip}
Find Worst Case Pull-Up Path Subcircuit
302
Find One Margin $V_{mar(1)}$
303
Find Worst Case Pull-Down Path Subcircuit
304
Find Zero Margin $V_{mar(0)}$
305
306
END

Fig. 3

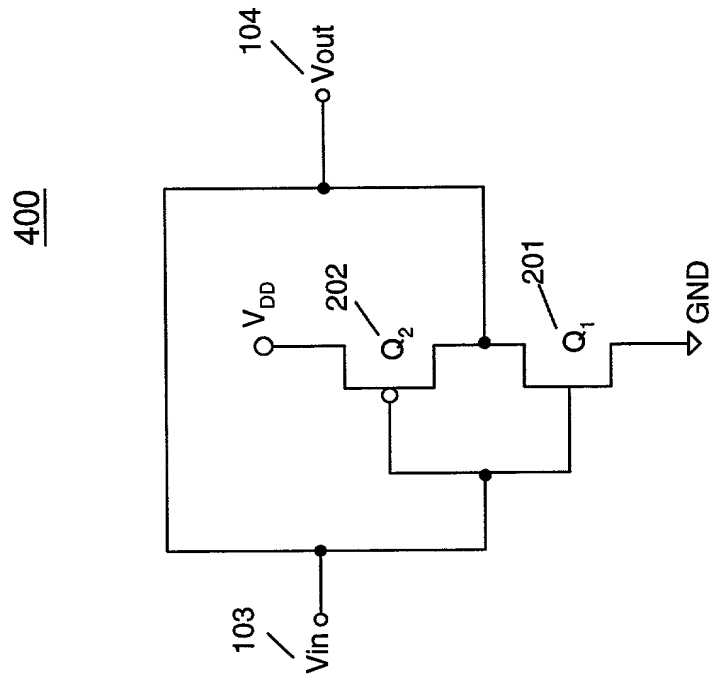


Fig. 4

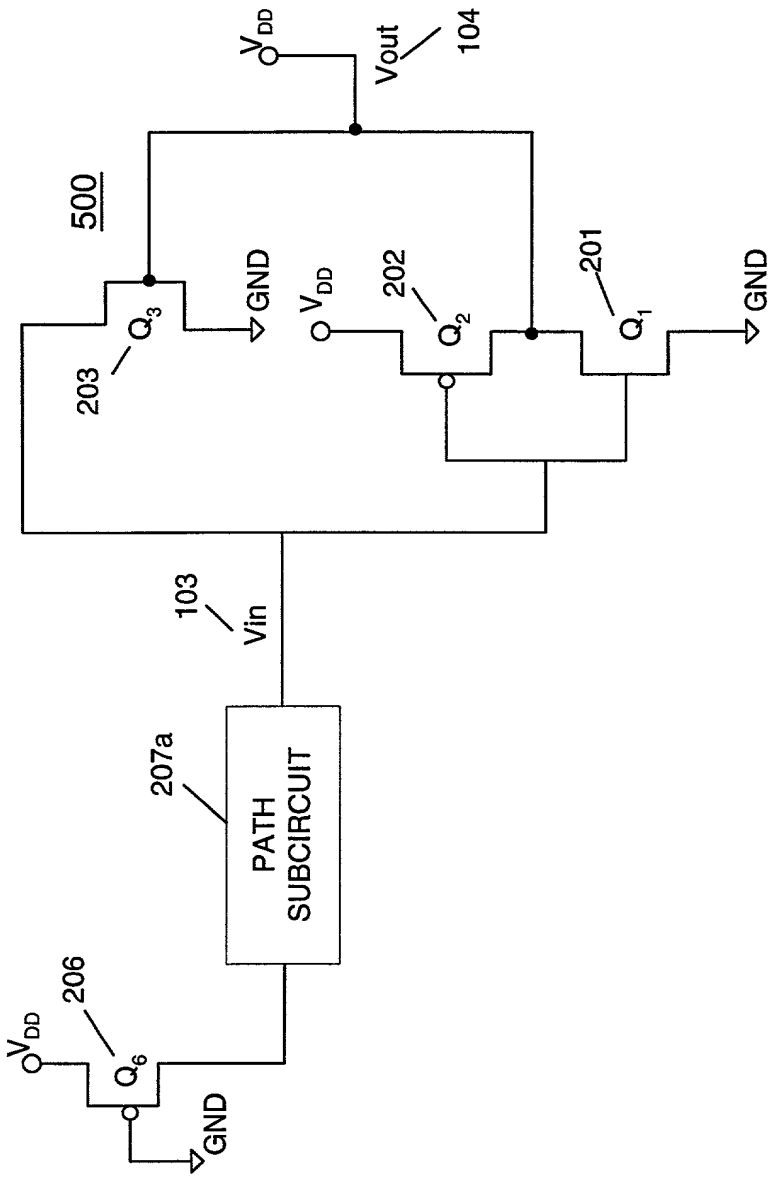

500
V_{DD}
Vout
104
GND
V_{DD}
202
Q_2
201
Q_1
GND
203
Q_3
103
Vin
207a
PATH
SUBCIRCUIT
206
V_{DD}
Q_6
GND

Fig. 5

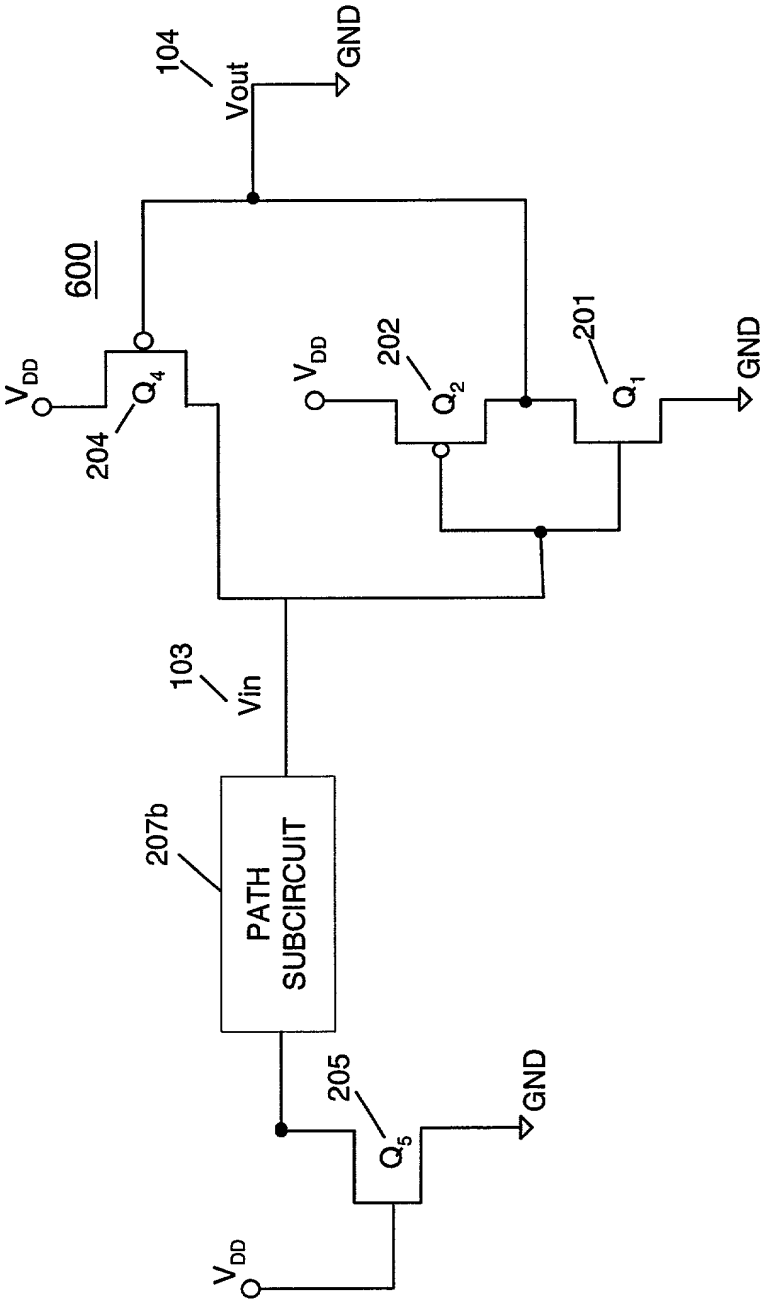

600
104
Vout
GND
V_{DD}
Q_4
204
V_{DD}
202
Q_2
201
Q_1
GND
103
Vin
207b
PATH
SUBCIRCUIT
205
Q_5
GND
V_{DD}

Fig. 6

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

ATTORNEY DOCKET NO. 10992563-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Method For Determining The DC Margin Of A Latch

the specification of which is attached hereto unless the following box is checked:

( ) was filed on _____ as US Application Serial No. or PCT International Application Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
N/A			YES: ___ NO: ___
			YES: ___ NO: ___

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
N/A	

U. S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS (patented/pending/abandoned)
N/A		

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Customer Number 022879

Place Customer Number Bar Code Label here

Send Correspondence to:
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80528-9599

Direct Telephone Calls To:
Alexander J Neudeck
(970) 898-4931

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Ted Scott Rakel **Citizenship:** US

Residence: 4924 Switchgrass Court Fort Collins CO 80525

Post Office Address: Same as residence

Ted Scott Rakel
Inventor's Signature

Jun 14 2000
Date

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION (continued)

ATTORNEY DOCKET NO. 10992563-1

Full Name of # 2 joint inventor: Douglas S Stirrett Citizenship: US

Residence: 1640 Kirkwood Dr #Q46 Fort Collins CO 80525

Post Office Address: Same as residence

Douglas S Stirrett 6/13/00

Inventor's Signature Date

Full Name of # 3 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

Full Name of # 4 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

Full Name of # 5 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

Full Name of # 6 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

Full Name of # 7 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

Full Name of # 8 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature Date

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

Application deficiencies were found during scanning:

☐ Page(s)_____ of _____ were not present for scanning. (Document title)

☐ Page(s)_____ of _____ were not present for scanning. (Document title)

Appendix is part of specification

☐ Scanned copy is best available.